COS 533: Advanced Cryptography — Princeton University
Lecture 23 (April 22, 2021)
Lecturer: Mark Zhandry — Scribe: Vinny Pagano

Notes for Lecture 23

1 Grover's Algorithm: Preliminary Setup

In this lecture, we will wrap up the cryptanalysis section. As it turns out, the period-finding algorithm for integer factorization and solving the discrete logarithm problem isn't quite the end of the picture. Namely, we shall introduce Grover's algorithm and discuss its advantages, limitations, and implications.

1.1 Preliminary Formulation of Problem

Problem: Consider a function $f : [N] \to \{0, 1\}$, and suppose for now that we have a promise that there is exactly one $x \in [N]$ such that $f(x) = 1$.

Goal: Find x.

Classically, we can easily solve this problem in $O(N)$ time via a brute-force search, where we try every single input. Note that this is the best we can do generically; if we know nothing about the structure of f and are limited to querying f on various inputs, then in expectation we see that

$$\begin{aligned}\mathbb{E}[\text{number of queries to find } x] &= \sum_{i=1}^{N} \Pr[\text{guessing } x \text{ after } i \text{ queries}] \\ &= \sum_{i=1}^{N} \frac{i}{N} \\ &= \frac{1}{N} \cdot \frac{N \cdot (N+1)}{2} \\ &= \frac{N+1}{2}.\end{aligned}$$

As we will discuss later, Grover's algorithm can solve this problem in $O(\sqrt{N})$ time.

We now present a more generalized version of the problem, which is followed by a description of the algorithmic procedure.

1.2 Generalization of Problem

Problem: Consider a function $f : [N] \to \{0, 1\}$, and suppose there exist exactly r distinct values $x_1, \ldots, x_r \in [N]$ such that $f(x_i) = 1$ for $i \in [r]$.

Goal: Find x_i for some $i \in [r]$.

As before, we can solve this problem classically in $O(N/r)$ time via a brute-force search, and this is the best we can do generically. However, Grover's algorithm can solve this problem in $O(\sqrt{N/r})$ time.

Remark 1 *Note that we don't need to know r. The procedure outputs a random x_i.*

2 Why is Grover's Algorithm Important?

For one, we get a quadratic speed-up over brute-force search for all NP problems. Why is this the case? Well, for any instance of an NP problem we can phrase it as a function f, where the input to f is a witness. We can do a brute-force search to find the witness and we will have

$$\text{run-time} \propto \text{length of witness.}$$

For some problems the brute-force method is the best classically, but Grover's algorithm implies that quantum mechanically we can improve our run-time. NP presumably remains hard for quantum computers, but for cryptography what this means is that for many constructions[1] in symmetric or secret key cryptography the best known algorithm is brute-force. The classical key sizes are set so that the best attacks take time $2^{128}, 2^{256}$, etc. typically. What Grover's algorithm means is that now the best attacks run in time $2^{64}, 2^{128}$, etc. which are twice as good as the best classical attacks. To attain an equivalent security to what we have in the classical case, we must double our key sizes. If some key has length 128 and we conjecture that the best security level from classically generated attacks is 2^{128}, we are not guaranteed security from a quantum attack using Grover's algorithm. In order to defeat its attack you have to increase the security level to 2^{256}. Hence, Grover's algorithm is important from a practical perspective and for cryptography in a quantum world.

3 Explanation of Grover's Algorithm

Start with a uniform superposition denoted

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N-1} |x\rangle.$$

[1]Consider NIST standardized hash functions, for instance.

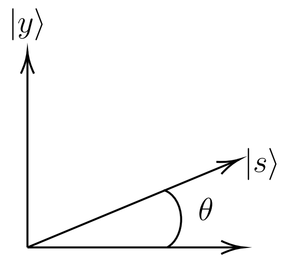

Figure 1: Initial setup of Grover's algorithm.

If we think of N as a power of 2, we can obtain this by applying Hadamard to the all-zero string $|0\rangle$ as we did for Simon's algorithm[2].

3.1 Preliminary Problem

First, consider the case where we have a unique solution y. We only need to consider two dimensions: one spanned by the unique solution and the other being $|s\rangle$. Note that the latter is a vector which is not quite orthogonal to $|y\rangle$, since their inner product is $1/\sqrt{N}$. We consider space spanned by these two vectors.

What our algorithm does is the following. Let θ denote the angle between $|s\rangle$ and the x-axis as shown in Figure 1. Note that θ can be computed explicitly using trigonometry. So what Grover's algorithm does is gradually move from $|s\rangle$ to $|y\rangle$ in steps of angle 2θ (in each iteration). The point is that θ is $O(1/\sqrt{N})$ in the asymptotic limit as N is large, but the angle between $|s\rangle$ and $|y\rangle$ is a constant less than $\pi/2$ radians, so the number of iterations is $O(\sqrt{N})$.

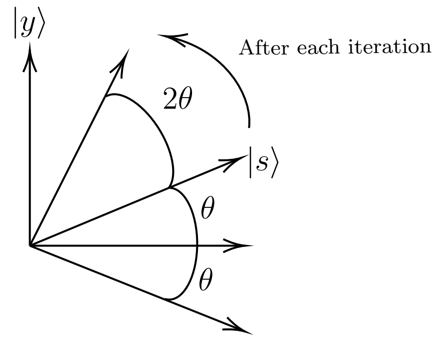

Figure 2: Grover's algorithm.

Main idea: if we can jump 2θ in every step, then the number of iterations is $O(\sqrt{N})$. Let's look at what one iteration does. Each iteration is going to have two steps:

[2]In general we may apply the Quantum Fourier Transform over the appropriate modulus to the $|0\rangle$ to get $|s\rangle$.

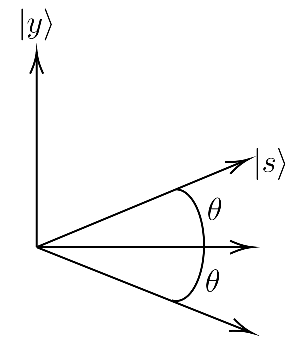

Figure 3: Grover's algorithm after step 1.

1. **Step 1.** In terms of standard basis vectors, we will map $|y\rangle \mapsto -|y\rangle$ and $|x\rangle \mapsto |x\rangle$ for all $x \neq y$. In other words, all terms in the quantum state will stay the same except the component in the y-direction, which changes sign.

   (a) Phase kickback. If we have state $\sum_x \alpha_x |x\rangle$, we prepare a new qubit

   $$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

   and append it to the system. This is Hadamard applied to $|1\rangle$.

   (b) Apply U_f, our unitary use[3] of f that we derive from f:

   - When x is not y, the unitary will XOR the output of f into the response system. By construction this is 0 so nothing happens.
   - When x is y, $f(y) = 1$ and when we XOR it with the state and we get a minus sign. Hence for this case we return

   $$-\alpha_y |y\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

   Therefore, in total we have

   $$\sum_x \alpha_x |x\rangle = \sum_x \alpha_x |x\rangle (-1)^{f(x)} \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

   We introduce a minus sign for all x where $f(x) = 1$. We can discard many qubits. So we turn our unitary into a phase unitary which uses f to determine the phase. So in terms of the picture, everything orthogonal to $|y\rangle$ doesn't change, the one orthogonal to $|y\rangle$ reflects along the x-axis after step 1 since we negate the component.

[3] Note that f is not invertible, so to turn it into something invertible we apply a unitary that has the input state and output state.

At a first glance this seems bad since we made negative progress, but we now combine this with step 2, which will get us back on the right track.

2. **Step 2.** We perform a reflection around $|s\rangle$.

   (a) First, we apply the Hadamard transformation $H^{\otimes n}$ (or QFT). We have:

   $$\begin{aligned} H^{\otimes n}|s\rangle &\to |0^n\rangle, \\ H^{\otimes n}|\psi\rangle &\to |\phi\rangle, \text{ where } |\psi\rangle \perp |s\rangle \text{ and } |\phi\rangle \perp |0^n\rangle. \end{aligned}$$

   (b) Send $|0^n\rangle \mapsto |0^n\rangle$ and $|x\rangle \mapsto -|x\rangle$ for $x \neq 0^n$.
   Anything orthogonal to $|0^n\rangle$ picks up a minus sign, and the component in the direction of $|0^n\rangle$ stays the same.

   (c) Apply $H^{\otimes n}$ again and note that $H \cdot H = id$. So under $H^{\otimes n}$ we have:

   $$\begin{aligned} H^{\otimes n}|0^n\rangle &\to |s\rangle, \\ H^{\otimes n} - |\phi\rangle &\to -|\psi\rangle, \text{ where } |\psi\rangle \perp |s\rangle \text{ and } |\phi\rangle \perp |0^n\rangle. \end{aligned}$$

   Observe that from (2b) the $|\phi\rangle$ in (2a) picks up a minus sign.

So we accomplish exactly what we wanted! This is the algorithm up to some subtleties.

For the algorithm described above, we denote the result of the nth iteration by $|s_n\rangle$, where $|s_0\rangle := |s\rangle$. So for $n > 0$ starting the algorithm with $|s_n\rangle$ (instead of $|s_0\rangle$) allows us to iterate the procedure.

3.2 Generalization of Problem

If there are many solutions as in subsection 1.2, we can change the y-axis to become a uniform superposition over all solutions

$$\frac{1}{\sqrt{r}} \sum_{y\,:\,f(y)=1} |y\rangle.$$

Note that the starting vector is still $|s\rangle$. The exact same phenomenon happens, except we have $\theta \approx 1/\sqrt{N/r}$.

3.3 Technicalities of Grover's algorithm

- **Why can't we reflect around $|s_1\rangle$?** The motivation behind this question is that, if we can do this, then we would make even more progress, making the algorithm even faster. The problem would be how to implement the reflection.

We can do it around $|s\rangle$ by Hadamard transformation, which is a constant time operation. On the other hand, in order to reflect around $|s_1\rangle$ we need a map from $|s_1\rangle \to |0^n\rangle$. But this map itself requires running one iteration of Grover's algorithm. More generally, reflecting on $|s_n\rangle$ requires n iterations. Stated simply, there is no easy way to reflect around a vector besides computing it from scratch. This is not an issue with the analysis or the algorithm. There is a bound that says that, if f is given as a black box, meaning we have oracle access to the unitary that computes U_f, then we need at least $\Omega(\sqrt{N})$ iterations.

- **What happens if θ does not divide $\pi/4$ radians?** If this happens, then we never hit $|y\rangle$ exactly. We will eventually loop around if we keep applying iterations of Grover's algorithm. Therefore, as a first pass we need to know when to stop, and we are never going to be exactly at $|y\rangle$. The number of iterations we need is $\lceil \pi/4\theta \rceil$. If we measure our state we will get $|y\rangle$ with probability $1 - O(1/\sqrt{N})$ because $\theta \approx 1/\sqrt{N}$.

- **What happens if we don't know r?** In order to move our state we need to know r, and we might overshoot and wrap around. We need to stop the algorithm at the right point. While in general we do not expect to know r, we can try to find it. To achieve this, run Grover's algorithm assuming r is a sufficiently high power of 2 and halve r after each iteration. One of these iterations will be within a factor of 2 of the correct r, which is close enough. Observe that N/r is an increasing power of 2, and in a geometric series the last term dominates. Hence, once you hit the right r, the running time will be determined by that r. This means that we can find r and the overall running time will still be $O(\sqrt{N/r})$.

4 Quantum Cryptography

As we have seen in previous lectures, post-quantum cryptography is classical cryptography which is secure under quantum attacks. (A rather unfortunate naming convention.) Quantum cryptography, on the other hand, concerns itself with actual quantum protocols. We like these since, as we have already seen, using the strange features of quantum computing allows us to achieve things that were once not possible. Chronologically, the first instance of quantum cryptography is through the concept of quantum money.

4.1 Quantum Money

Recall that we alluded to quantum money in a previous lecture as an application of no-cloning. Define 4 states $|\phi_{b,c}\rangle = H^b|c\rangle$ for $b, c \in \{0,1\}$.[4] We know by no-cloning that it is impossible to clone $|\phi_{b,c}\rangle$ for unknown b and c because these four vectors are not orthogonal. (Recall that if you have non-orthogonal vectors you cannot clone.)

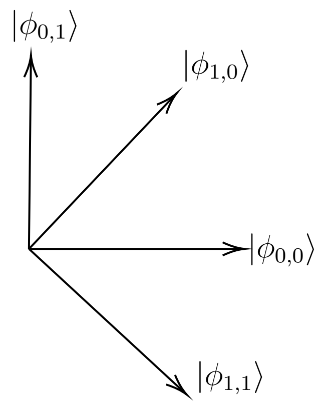

Figure 4: Pictorial setup for quantum money scheme.

For the quantum money scheme, we have a banknote. In particular, we consider a serial number (b, c) and a note $|\phi_{b,c}\rangle$. To verify the banknote against the serial number, we take as input the serial number and a supposed state $|\phi\rangle$ that may or may not be the right state. We define Verify as follows.

$\mathsf{Verify}((b,c), |\phi\rangle)$ will:

- measure $H^b|\phi\rangle \to |c'\rangle$.
- output 1 (accept) if and only if $c' = c$.

On correctness and security of quantum money scheme

Note that $H^b|\phi_{b,c}\rangle = |c\rangle$. Thus, if we were to set $|\phi\rangle = |\phi_{b,c}\rangle$ so that we perform $\mathsf{Verify}((b,c), |\phi_{b,c}\rangle)$, then when we measure we get $|c\rangle$ back and our note is correctly accepted. However, we shall show that security of our scheme is currently weak.

Suppose an adversary is given $|\phi_{b,c}\rangle$ but not (b, c). They try to split the note into two possibly entangled states $|\phi_1\rangle, |\phi_2\rangle$. If we run Verify on both, getting b_1 and b_2, then

$$\Pr[b_1 = b_2 = 1] = 5/8.$$

This is a consequence of no-cloning. One attack is you make it guess b' for b, measure $H^{b'}|\phi_{b,c}\rangle \to c'$, and then construct the state $|\phi_{b',c'}\rangle$ twice and this is what it outputs.

[4]Under this notation, we apply Hadamard if and only if $b \neq 0$.

If $b' = b$ (occurs with probability 1/2), the adversary wins with probability 1 since they recover c exactly. If $b' \neq b$, then c' is random, and the vector they get is at a 45 degree angle from the right quantum state. And what this means is that each Verify will pass with probability 1/2. So the overall attack probability is

$$\Pr[b_1 = b_2 = 1] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \left(\frac{1}{2} \cdot \frac{1}{2} \right) = \frac{5}{8}.$$

Remark 2 *One can slightly improve this to* 3/4.

The main point is that there is a constant probability of attack, so this is not a particularly good money scheme since there is a constant probability that you can clone. However, we can boost security by modifying our quantum money scheme.

Boost security: Repeat many times so your serial number is a vector

$$\vec{b} \in \{0,1\}^n, \vec{c} \in \{0,1\}^n, |\phi_{c,b}\rangle = |\phi_{b_1,c_1}\rangle |\phi_{b_2,c_2}\rangle, \ldots$$

Remark 3 *It can be shown that under this modified scheme the best attack has probability C^n for some constant $C \in (0,1)$, which is exponentially small.*

This concludes the basic quantum money scheme.

5 Next Time

What remains to be discussed are the limitations of quantum money. Specifically, the fact that the serial number needs to be secret. We shall also review some extensions. Essentially, there is a whole world out there of potential cryptographic applications that rely on the unclear inability of quantum states, which is something that we don't have classically; and this, in principle, can be used to realize all sorts of novel cryptographic applications. This will be the topic for the next time.